

Batch Modernization: Batch Improvements in z/OS 1.13/2.1

Presentation created by: Gary Puchkoff
Speaker Company: IBM Corporation

Date of Presentation: March 19, 2013

- This presentation represents function shipped in V1.13
- This presentation presents function announced in the z/OS 2.1 preview announcement of February 5th 2013.
 - Likely syntax and functional usage are presented in this presentation.
 - IBM reserves the right to change the implementation specifics prior to announcement.
 - No warranty is made regarding specific details of V2.1 presented in this presentation

Agenda

IBM' s Batch Modernization effort

- Batch Runtime Environment
- Improvements to z/OS
 - V1.13
 - V2.1 preview

z/OS Batch – Integral part of the OS

- OS based batch submission
 - Always there
 - Scheduled
 - Secure
 - Resource Accounting
- OS based resource management
 - Dataset synchronization via enqueues
 - CPU, memory, I/O
- OS based device management
 - Disk, Tapes

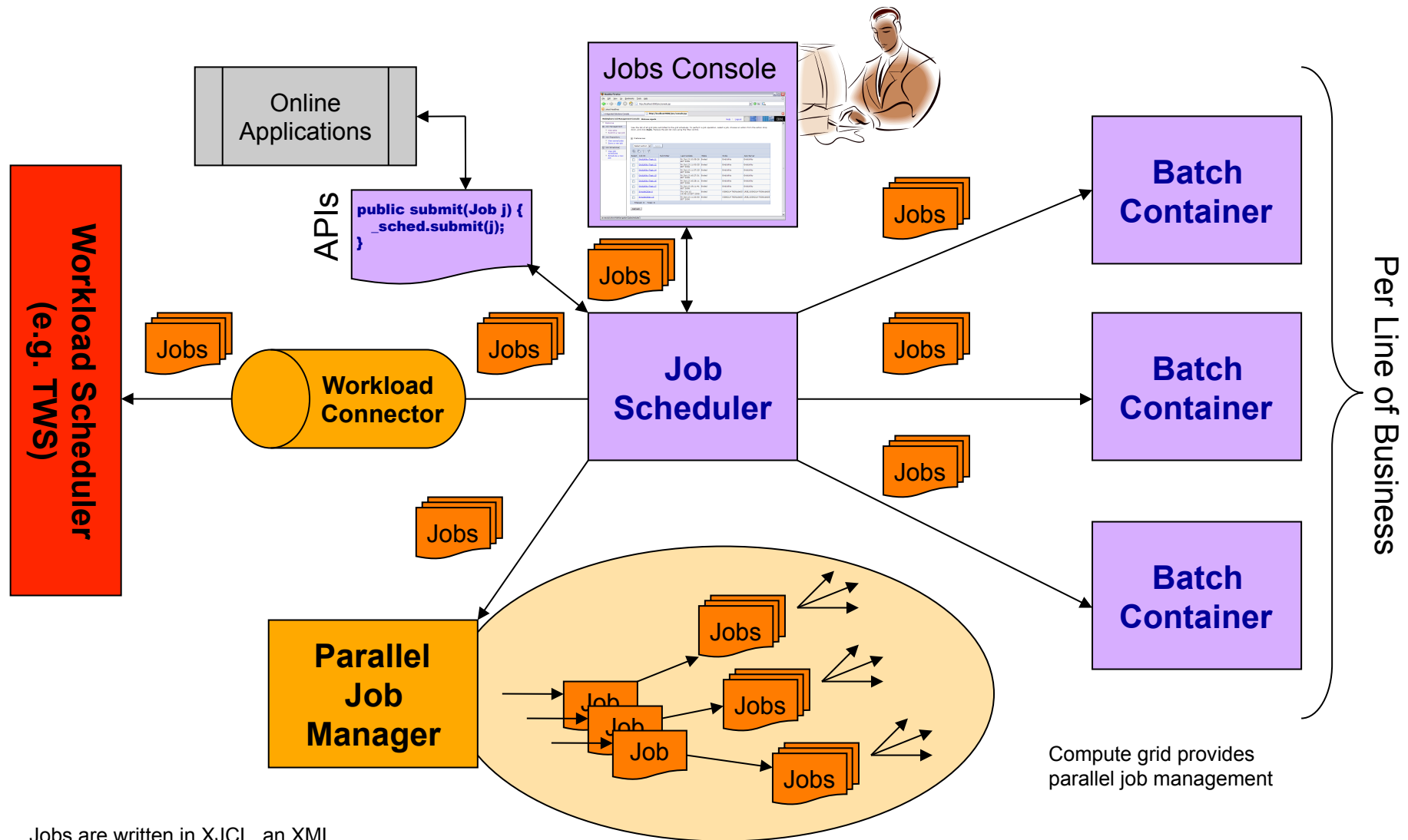
IBM Batch Modernization initiative

- Ensure that IT architects and the industry recognize
 - Almost all large scale IT projects include batch
 - The web focus of the last decade has focused on online work, response time, transactions
 - Backend processing is often best implemented as batch
- Provide an IBM Batch solution for any platform
 - Including z/OS
 - Websphere Compute Grid
 - Both java programming model and execution environment
- CICS Compute Grid on z/OS
- Maintain leadership in z/OS batch
 - Improvements to existing function
 - Integrate and support the Compute grid programming model

WebSphere XD Compute Grid

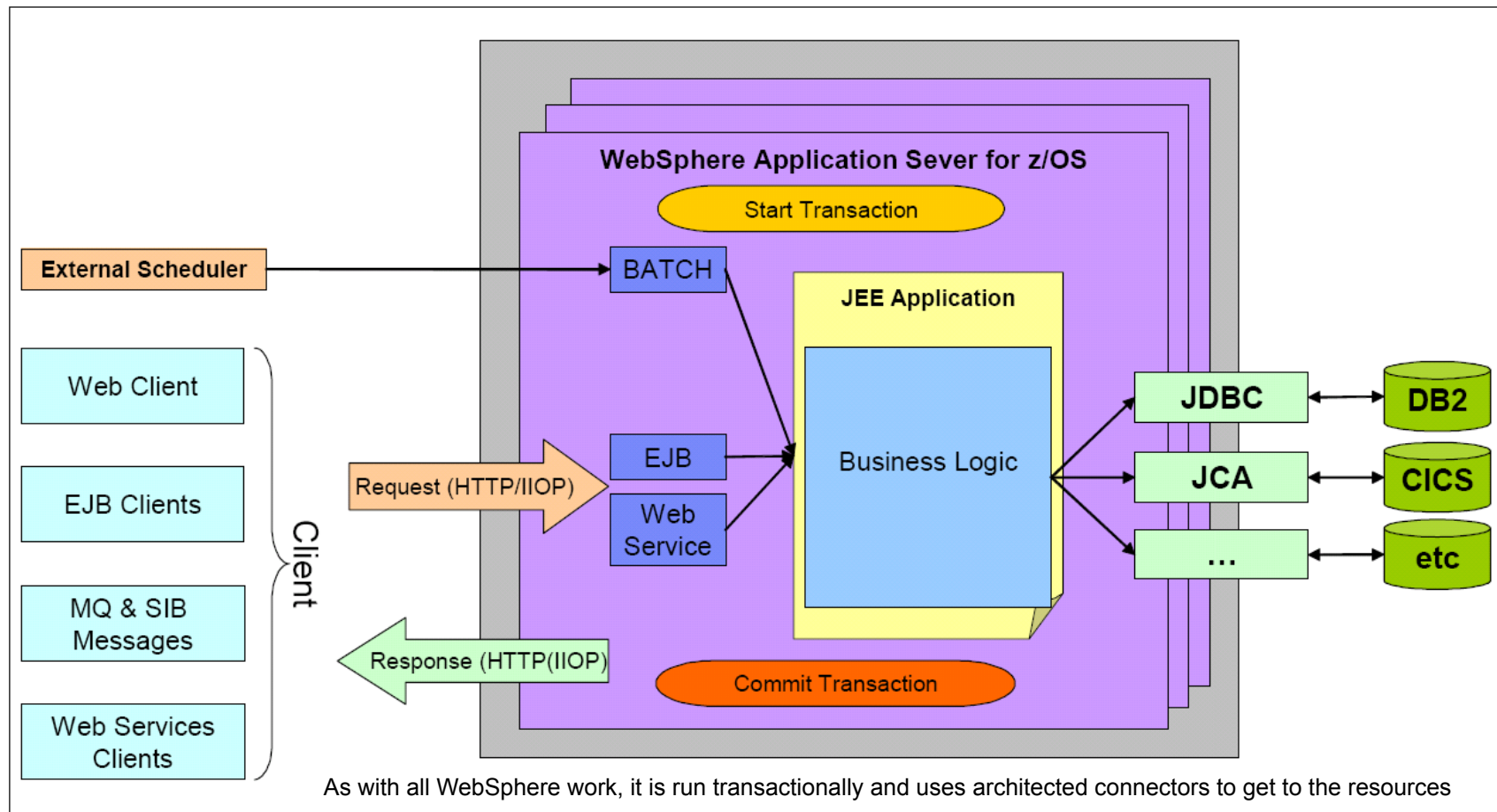
- Java Batch combined with all functionality provided by the WebSphere JEE Container
- Extension on top of WebSphere Application Server
 - Available as a feature pack on WAS V7 (limited function), or a product, WebSphere XD Compute Grid (full function)
- Java stand-alone functionality **plus**
 - WAS Container management
 - security, transactions and connection management
 - Check pointing
 - Persistent JVM
 - QoS, such as high availability
- **Reuse of OLTP** code in WAS XD Compute Grid Batch Container

Compute Grid Overview diagram



Jobs are written in XJCL, an XML version of a job control language.

WebSphere XD Compute Grid programming environment



Benefits of Java (Batch) on z/OS

- Specific Java Batch APIs for z/OS
 - Dataset and VSAM access
 - Condition Code passing
 - DFSORT support
 - Writing Logstreams
 - Triggering of Jobs from Java
 - RACF APIs
- Local DB2 Database driver for **high throughput**
- Access to many Java skills
- Effective and efficient **development tools** available
 - Rational Tools available specific to Batch Container
- Availability of many classes, libraries, frameworks and applications based on open source
- Interoperability with other programming languages on z/OS

z/OS Batch Runtime Environment (V1.13)

- A new option for running batch work in z/OS
- Provides a managed environment for integration of Java, COBOL, and PL/I (new v2.1)
- Consistent with IBM Websphere based batch
 - A subset of the Websphere programming model
 - Incorporated in the OS
- DB2 and tVSAM (new v2.1) resource managers

Batch Execution Runtime Environment

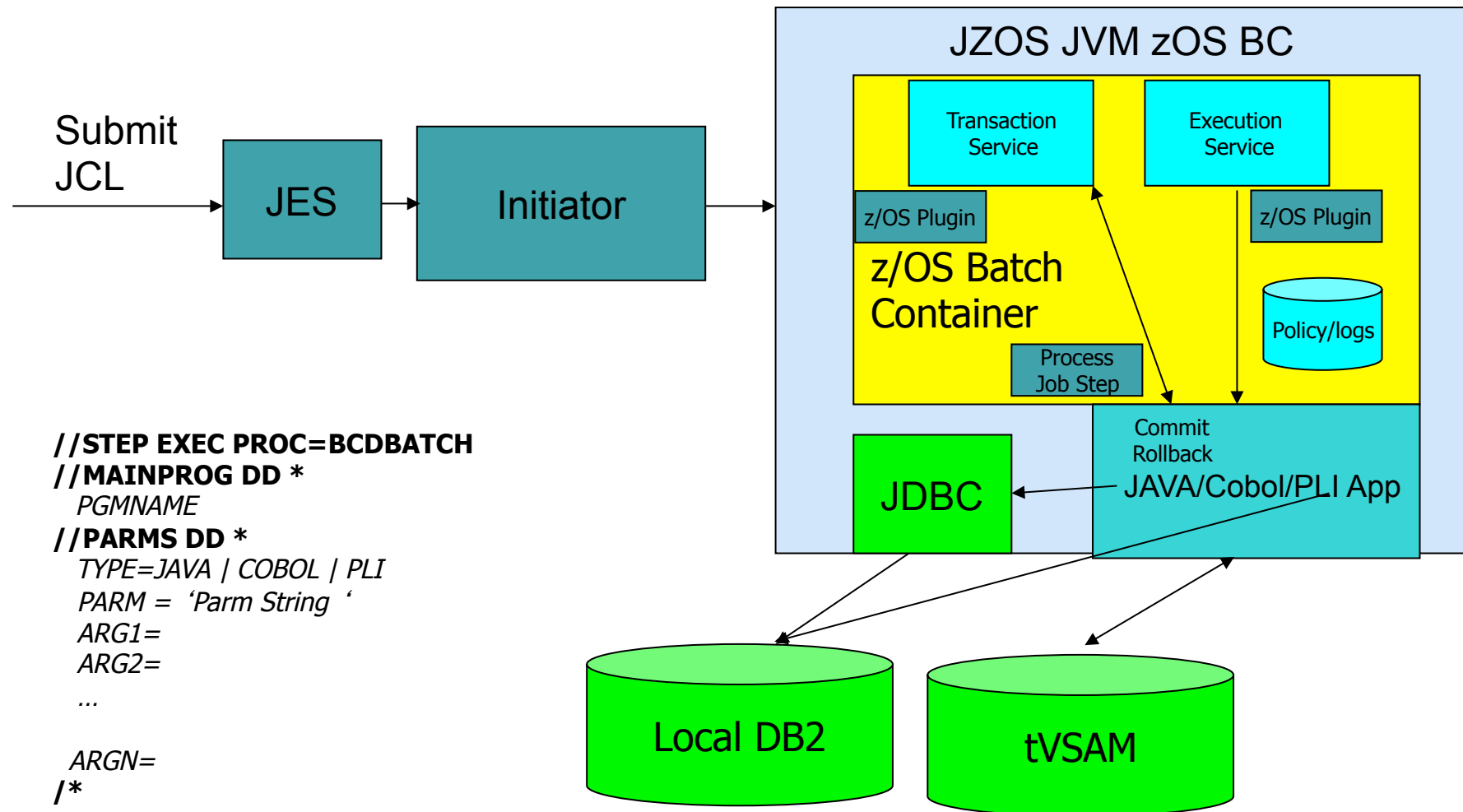
Java COBOL PLI with DB2 and tVSAM

Interoperability

- Ability to replace/add functions in current 3GL DB2 (e.g. COBOL DB2) application inventory with new Java DB2 code
 - Requires local attach z/OS DB2 connection sharing for common DB2 access
 - Requires UOW (Transactional) integrity among the application components
- A generalized solution without requiring a specific run-time or middleware, i.e. a pure batch environment
- Implementation requires little or no changes to existing code!
 - Only requires special callbacks for commit/rollback

Our z/OS Topology

JES Single Step based



JCL is familiar to operations
code in the batch container is the same as the code one could write anywhere

z/OS Batch Runtime Environment

- Consistent with the WAS Batch Facility
 - Batch programming model consistency (not all of WAS)
 - xJCL available if necessary (v2.1)
- Runs single LE enclave per step
 - Each step will require startup of java again
 - Vs WAS which creates new enclaves as needed
- Can run java, cobol to java to cobol again
 - Requires cobol code to run 're-entrant'
- Share static COBOL connection with JDBC dynamic Java connection

JOBS REST Interface (New)

- Improve access to batch
 - Increase client coverage
 - Add a non-controversial remote interface
 - Align with Web technologies
 - Make z/OS a little more approachable
- A new programmatic interface using HTTPs
 - Delivered in z/OSMF
 - Accessible both locally and remotely
- Submit JCL, get status, retrieve output files, change jobclass, cancel job, purge job, submit from file or dataset (V1.13 SPE)
- Asynch. notification, hold job, release job, submit to secondary, submit with symbols including job correlator (V2.1)

JOBS REST Interface (cont)

- Today' s options
 - Allocate and open internal reader
 - TSO/ISPF submit,
 - FTP “interface-level2”
 - Java z/OS submit interface
- New option HTTPs based API
 - Secure
 - Firewall friendly
 - Text like

JOBS REST Interface (cont)

- Example
 - A job named G1JAVA1B with job number JOB00023

GET **/zosmf/restjobs/jobs/G1JAVA1B/JOB00023** HTTP/1.1

HTTP/1.1 200 OK

Date: Thu, 13 Jan 2011 05:39:28 +0000GMT

Content-Type: application/json

Connection: close

```
{
  "jobid": "JOB00023", "jobname": "G1JAVA1B", "subsystem": null, "owner": "G1JAVA1",
  "status": "OUTPUT", "type": "JOB", "class": "A", "retcode": "CC 0000",
  "url": "https://host:port/zosmf/restjobs/jobs/G1JAVA1B/JOB00023",
  "files-url": "https://host:port/zosmf/restjobs/jobs/G1JAVA1B/JOB00023/files"
}
```

Using the files-url you can retrieve the individual datasets, you can retrieve the data in chunks etc.

Instream data in PROCs and INCLUDEs

- Instream data in procs and includes

```
//PROC1 PROC  
//ASTEP EXEC PGM=xyz  
//DD1 DD *  
    This is instream data  
//      PEND
```

- Prior to this support you could not have instream data in a JCL procedure
- Support also allows instream data in include statements wherever a DD statement is allowed
- For JES2 requires both z/OS 1.13 and JES2 1.13 on the converting system and the initiating system
- For JES3 requires z/OS 2.1 and JES3 2.1
- This is not supported for MSTR subsystems

Instream data in PROCs and INCLUDEs

Instream data in PROC example

```
//HELLO    PROC
//STEP1    EXEC ASMHCLG
//C.SYSIN DD *
TEST      CSECT ,
          STM   14,12,12(13)
          BALR  12,0
          USING *,12
          ST    13,SAVAREA+4
          LA    13,SAVAREA
          SPACE 1
          WTO   'Hello world!'
          SPACE 1
          L     13,SAVAREA+4
          LM    14,12,12(13)
          SR    15,15
          BR    14
          SPACE 1
SAVAREA    DC   18F'0'
          END
//L.TEST   DD DUMMY
//L.SYSXX   DD *
//          PEND
```

Exporting Symbolics to runtime

- You can now have a JCL symbolic made available at runtime
- `// EXPORT SYMLIST=(variable1, variable2)`
- `// SET variable1=value`
- Use LE service xxx to retrieve symbols by name
 - `Var vstring init("variable1");`
 - `Val char[255];`
 - `Fc longint;`
 - `CEEGTJS(1,var,val,Fc);`
- Value of variable at end of step containing export during conversion is used

Instream symbolic substitution

- For JES2 V2.1 we add support for substitution in instream data

```
//PROC1 PROC
// SET VARA=HELLO
// EXPORT SYMLIST=(VARA)
//ASTEP EXEC PGM=xyz
//DD1 DD *,SYMBOLS=JCL
      This is instream data, &VARA
//      PEND
```

Results in:

This is instream data, HELLO

- Note: you have to export the variables referenced as they are substituted just before the buffer is handed to the reader.



Retrieving Exported Symbols

- CEEGTJS(Fn,symbolName, symbolValue, symValLen, Fc)
 - Where Fn is a fullword function code = 1
 - symbolName is vstring representing the name of the symbol
 - A vstring is a halfword length followed by the names characters
 - Lower case is folded to upper case
 - symbolValue is a 255 character input area that is filled with the symbolValue padded with blanks
 - symValLen is a fullword integer representing the non-blank length of the symbolValue
 - Fc is the feedback code of the function
 - CEE000 – success
 - CEE3L9 – symbol not found

System Symbols support for Jobs

- Any system symbol (IEASYMxx) can be referred to in JCL
- Will be resolved by the converting system during job conversion

//name JOB (Acct),programmer

//STEP1 EXEC PGM=ABC

//DD1 DSN=&SYSNAME.AAA.BBB,DISP=SHR

Care should be taken when system symbols vary across the sysplex as jobs can be converted on one system and executed on another.

Passing in Symbolics to JCL

- A new option on the internal reader for JES2
- Will specify what symbolics should be passed into a JOB
// SET DSNAME=A.B.C
//SYSUT2 DD SYSOUT=(A,INTRDR),
SYMLIST=(DSNAME)
- The value of these symbolics is evaluated at time of submit and passed into the JOB. They act as if they were coded on a SET statement immediately after the jobcard.
- Support is initially provided in the z/OSMF REST Jobs API to specify these



Job Correlator/notification (JES2)

- Two special symbols are provided
 - SYSJOBCCR – a 32 byte user selected string
 - Notification url
- Job correlator is a 32 byte string that the submitter of the JCL can provide
 - It should be unique and can be used to retrieve the job later
- Notification URL is a URL that can be posted when a job completes
 - It requires CEA and CIM to up and running
 - It requires a suitable server to be running to field the post



- **Long Parameters**

- **New PARMDD EXEC keyword support longer parameter strings**
 - Mutually exclusive with PARM keyword
 - No other changes required for unauthorized programs
 - Authorized programs must be re-linked using LONGPARM in order to support >100 bytes or system will terminate the job at step initiation
 - Supports parameter lists from 1 to 32,760 bytes long
 - DD can also point to uss file, PDS(e), or seq

Example:

```
//NOTAREAL JOB (accounting info),MSGLEVEL=(1,1),CLASS=BATCHLOW,  
// NOTIFY=&SYSUID  
//*  
//UNAUTH EXEC PGM=MYPGM, PARMDD=PARMS  
//IN      DD DISP=SHR,DSN=MY.DATA.SET  
//OUT     DD DISP=(,CATLG),DSN=MY.NEW.DATA.SET, ...  
//PRINT   DD SYSOUT=*  
//PARMS   DD *  
LONG PARAMETER LIST HERE IN THE DATA SET NAMED BY  
PARMDD.  NOTE THAT IT NEED NOT BE AN INSTREAM DATA  
SET.  A SEQUENTIAL DATA SET OR A MEMBER OF A PDS OR  
PDSE WILL WORK AS WELL.  AND, IF I COUNTED RIGHT,  
THEN THIS VERY VERY LONG PARAMETER LIST IS WELL  
OVER 100 CHARACTERS IN LENGTH AND I CAN STOP TYPING!  
/*
```

8 Character Job Classes

- **8-character Job classes**

- Currently, CLASS on the JOB JCL statement is a single alphanumeric character, so the maximum number of job classes that can be specified is 36
- JES3 supports 8-character job classes via JECL (/*MAIN CLASS=xxxxxxx)
 - JES2 does not
 - JES3 continues to override CLASS from the JOB statement if CLASS is also coded on the /*MAIN statement
- The JCL JOB statement will be modified to support 8-character job classes using alphanumeric characters
- JES3 will continue to support a maximum of 255 job classes
- No explicit limit planned for JES2
- Both JES' s optionally will check a new SAF resource for JESJOBS
 - JOBCCLASS.*nodename.jobclass.jobname*
- The larger JOBCCLASS will show up in an extended JMR, and SMF records

JOB return code (JES2 only)

- There is no formal definition of the job return code. The defacto standard is JES2 returning the maximum return code of the job.
- New job card keyword to control job return code
 - `JOBRC= MAXRC | LASTRC | (STEP,name.name)`
 - MAXRC is existing processing (default)
 - LASTRC is return code of last step
 - (STEP,*name.name*) is return code of identified step
 - *If step not executed, defaults to MAXRC*
- `JOBCLASS JOBRC= MAXRC|LASTRC` to affect processing for all jobs in the job class
- Affects return code seen in
 - Extended status (eg SDSF)
 - ENF 70
 - HASP165 message
 - `$DJ,CC=` command



Suspend a job at next step (JES2 only)

- If a job is restartable
 - A new operator command is defined to suspend the job at the next step
 - The job is requeued for execution
- Allows a faster and/or less disruptive shutdown
- The job must complete the step it is running.
 - This remains an issue for long running steps



Suspend a job at next step (JES2 only)

- Remove job on step boundary
 - New STEP operand on \$EJ command
 - Causes job to exit execution at end of current step
 - Optional HOLD operand makes job held
 - Job is queued for execution
 - Job must be journaling (JOURNAL=YES on JOBCLASS)
 - Uses existing continue restart function of z/OS
 - Previously used to restart jobs after an IPL
 - Full syntax \$EJxxx,STEP [,HOLD]
 - Full cross member support



Spin Any (JES 2 only)

- Ability to specify automatic spin options in JCL
 - Eliminates the requirement to take down a long running process to release spool space
 - Similar to what was done for JESLOG
 - Applies to any data set allocated as SPIN
 - No application code/JCL change needed
 - Provides a time, size, or command based option to spin output

- Update to SPIN= parameter on DD statement

SPIN= { NO }	
{ (UNALLOC,'hh:mm') }	Spin at specific time
{ (UNALLOC,'+hh:mm') }	Spin every hh:mm interval
{ (UNALLOC,nnn [K M]) }	Spin every nnn lines
{ (UNALLOC,NOCMND) }	Cannot be spun by command
{ (UNALLOC,CMNDONLY) }	Can be spun via operator command

- \$TJn,SPIN,DDNAME=*name* command added



New system/system affinity on JOB

- Two new keywords are supported on a job
 - SYSTEM=(system name list),
 - SYSAFF=(member name list),

This eliminates the need to code a JECL statement for this function

- In JES2 /* JOBPARM,SYSAFF=masname
- In JES3 /* MAIN SYSTEM=system

JCLLIB support for JES proclib (JES2)

- // JCLLIB ORDER=(dsname),PROCLIB=(ddname)
- Allows specifying the location of a named JES proclib after the JCLLIBs are searched.
- Replaces the need for
 - /* JOBPARM PROC=nn in JES2



OUTPUT JCL Changes (JES2)

- MERGE=YES
 - Allows an OUTPUT statement to be used as a template and merged with instance specific OUTPUT options
- DDNAME=reference
 - Allows an OUTPUT statement to be coded so that it applies to only specific SYSOUT instances
 - Searching is from most specific reference to least



Batch Parallel Recall

- **Batch Parallel Recall**
 - Allocation to determine whether data sets to be allocated have been migrated
 - For DFSMSHsm-migrated data sets, Allocation will
 - Issue a recall request for each one during step initiation
 - Wait for all recalls to complete
 - Continue with Allocation processing needed to start the step
 - New ALLOCxx keyword to enable, and SETALLOC support

Freeing Tape Volumes before end of step

- A new FREEVOL=EOV keyword on the JCL DD statement:
 - This is intended to allow overlapped processing for multivolume data sets, which can reduce batch processing elapsed time.
 - Specifies that a tape that is a part of a multivolume data set become available at end-of-volume rather than at step end.
 - Allows other jobs to use the tape immediately.
 - Allows overlapped processing of multivolume tape data sets.

IEBCOPY

- IEBCOPY performance is improved for copying from a partitioned data set (PDS) to another PDS
- IEBCOPY now exploits 31-bit storage for buffers
- The requirement for APF authorization of IEBCOPY is removed
- Fof V2.1 – Copygroup for PDS' s
 - As for PDSE copy aliases along with specified members automatically
 - PDS/PDS, PDSE, PDS, PDS/PDSE, PDSE/PDSE all work the same
 - PDS and PDSE unload work the same
 - PDS load not planned to support COPYGROUP as PDSE load does
 - Support for pattern matching
 - Using * and %

COPYCNT

- New JCL OUTPUT statement keyword
 - Also available through dynamic Output
- Logical synonym for COPIES=
- COPYCNT=x
 - Where x can be 0-2G
 - Usage depends on print driver... if the print driver is COPYCNT aware it will supersede any specification of COPIES=
- z/OS now allows the specification of copies >255
- No support in JES2 or JES3 for detecting or changing the value of COPYCNT, not reflected in any JES control blocks
- SDSF can display and change COPYCNT

Typrun=Scan

- JES2 will now optionally call Converter AND Interpreter back to back
- JES3 has always done this
- This means the right hand side of the = will be verified as well as the left hand side
- At this time it is optional for JES2
 - Implemented by a new CI address space like function
 - It does not do any catalog locates or verification of datasets



More coming in V2.1

- DFSMS – GDGs, PDSE, VSAM
- zFS – directories, size
- Better integration of errors found during JES2 input processing in JECL and JCL with converter processing
 - JECL cards assigned statement numbers
 - Error messages in normal system messages data sets
 - JECL errors do not stop calling converter to find other errors
- Processing of JESDS OUTPUT cards even when jobs do not make it into execution
 - Part of back to back C/I processing
 - Better output processing in the event of an error
- Optionally control where job converts based on scheduling environment
 - Ensures consistency of system symbols used in JCL
- Support to allocate internal reader from any address space
 - Do not need a JES2 environment (request jobid) to submit a job
 - Submit from address spaces that are SUB=master
- Etc.

Options for improved Batch Elapsed time

- Hardware
 - CPU
 - Faster – move up to zEC12 over z196 or z10, faster capacity model
 - More engines or specialty engines
 - I/O
 - zHPF – high performance ficon – improved i/o rates and latency
 - SSD drives – solid state disk – improved i/o rates and latency
 - Memory
 - Large pages(1M or 2G) – managed in larger chunks
- Software
 - Parallel
 - I/O, setup, execution
 - Optimize
 - Use less resource, exploit new algorithms

References

- Redbook – Batch Modernization for z/OS – SG24-7779
<http://www.redbooks.ibm.com/redbooks/pdfs/sg247779.pdf>
- Redpaper – Batch Processing with Websphere Compute Grid: Delivering Business Value to the Enterprise
<http://www.redbooks.ibm.com/redpapers/pdfs/redp4566.pdf>
- Java Batch Programming with XD Compute Grid
http://www.ibm.com/developerworks/websphere/techjournal/0801_vignola/0801_vignola.html

Question

- Where else would you like to see us focus on batch modernization?
- Please send e-mail to Gary Puchkoff
(puchkoff@us.ibm.com)

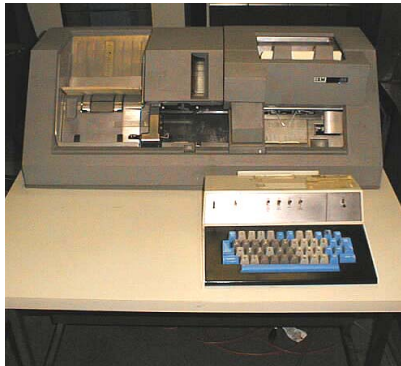
Backup

What is batch computing?

Batch processing system for processing data with little or no operator intervention. This allows efficient use of the computer and is well suited to applications of a repetitive nature, such as file format conversion, payroll, or the production of utility bills.

In **interactive computing**, by contrast, data and instructions are entered while the processing program is running.

Hutchinson Encyclopedia



<u>Batch / Bulk Processing</u>	<u>Interactive Computing</u>
Interface designed for machine	Interface designed for human
Optimized for machine utilization	Optimized for efficiency of human
Many items of data processed per human intervention	Single transaction processed per intervention
Automated	Manual
Served well by legacy languages & technology – COBOL, JCL, JES	Served well by modern languages & frameworks – Java, WebKit
Ceased to be a significant element of University curricula	Significant part of University curricula

To **make** a payment complete the details below and click 'Make payment'.
Please be aware that your name may show on the statement of the person you are paying.

To **delete** the recipient details from your list, click 'Delete recipient'.

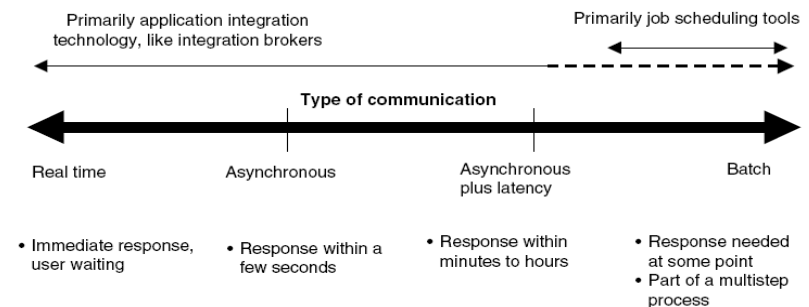
Pay to: BT OPENZONE
Sort code: 20-00-00
Account number: 00835757
Reference (if any): BTOZ_192852769

Amount: £ .

Date/Type: ☒ As soon as possible
☐ Future payment - - This can be up to 31 days in the future.

The time taken for this payment to reach the recipient will depend on the option you choose.
Find out more about [payment timescales](#).

- By eliminating human reaction time, batch can process in one hour what OLTP would take one month (or require 50,000 people)
- Batch is part of a continuous spectrum of workload
- Enterprise clients run a continual mix of online and batch – response to global, 24 x 7 business
- Business events as a trigger for batch e.g. end-of-day are a hallmark of a critical workload



Source: "Consider Scheduling Tools for Batch Application Integration", Gartner.

Why Batch?

- Economies of Scale
 - Processing all items in a collection
 - Pre-fetch optimizations
- Period based processing
 - End of day, month, year
 - Consistent reporting
 - Integration with others on a consistent basis

z/OS 45 years of Batch – From punch cards to Java Batch

45 years of evolution in Batch processing on the IBM Mainframe have provided the foundation for heavy-duty, reliable and efficient Batch for most large companies in the world:

- WLM, WLM Batch initiators
- [Batch & Print Subsystem](#) JES2, JES3, PSF
- Job Control Language (JCL)
- Batch Management Interfaces (for example. [SDSF](#))
- Step and Job dependencies by means of Condition Codes and [Job Networks](#)
- Online and Batch in parallel
- Time-driven Job execution
- Job / Step Restart functions, Start, Submit, Remote submit, Syntax Scanner
- [Accounting](#) based on Job/USER, Job statistics and RMF reports
- Pre-loaded address spaces (initiators)
- All Mainframe programming languages can be used in Batch

